Fast, Scalable Disk Imaging with Frisbee

University of Utah

Mike Hibler, Leigh Stoller,
Jay Lepreau, Robert Ricci,
Chad Barb

Key Points

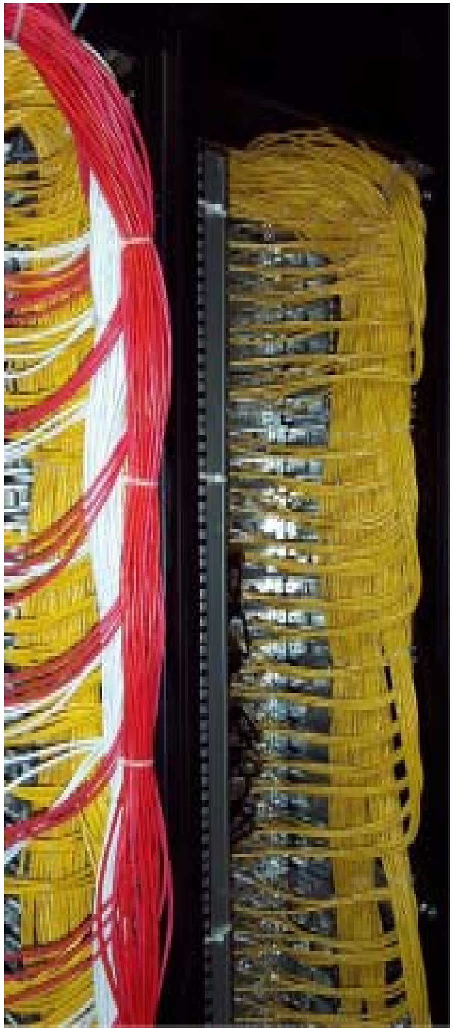
Frisbee clones whole disks from a server to many clients using multicast

- Fast
  - 34 seconds for standard FreeBSD to 1 machine
- Scalable
  - 34 seconds to 80 machines!
- Due to careful design and engineering
  - Straightforward implementation loaded in 30 minutes

Disk Imaging Matters

- Data on a disk or partition, rather than file, granularity
- Uses
  - OS installation
  - Catastrophe recovery
- Environments
  - Enterprise
  - Clusters
  - Utility computing
  - Research/education environments

Emulab



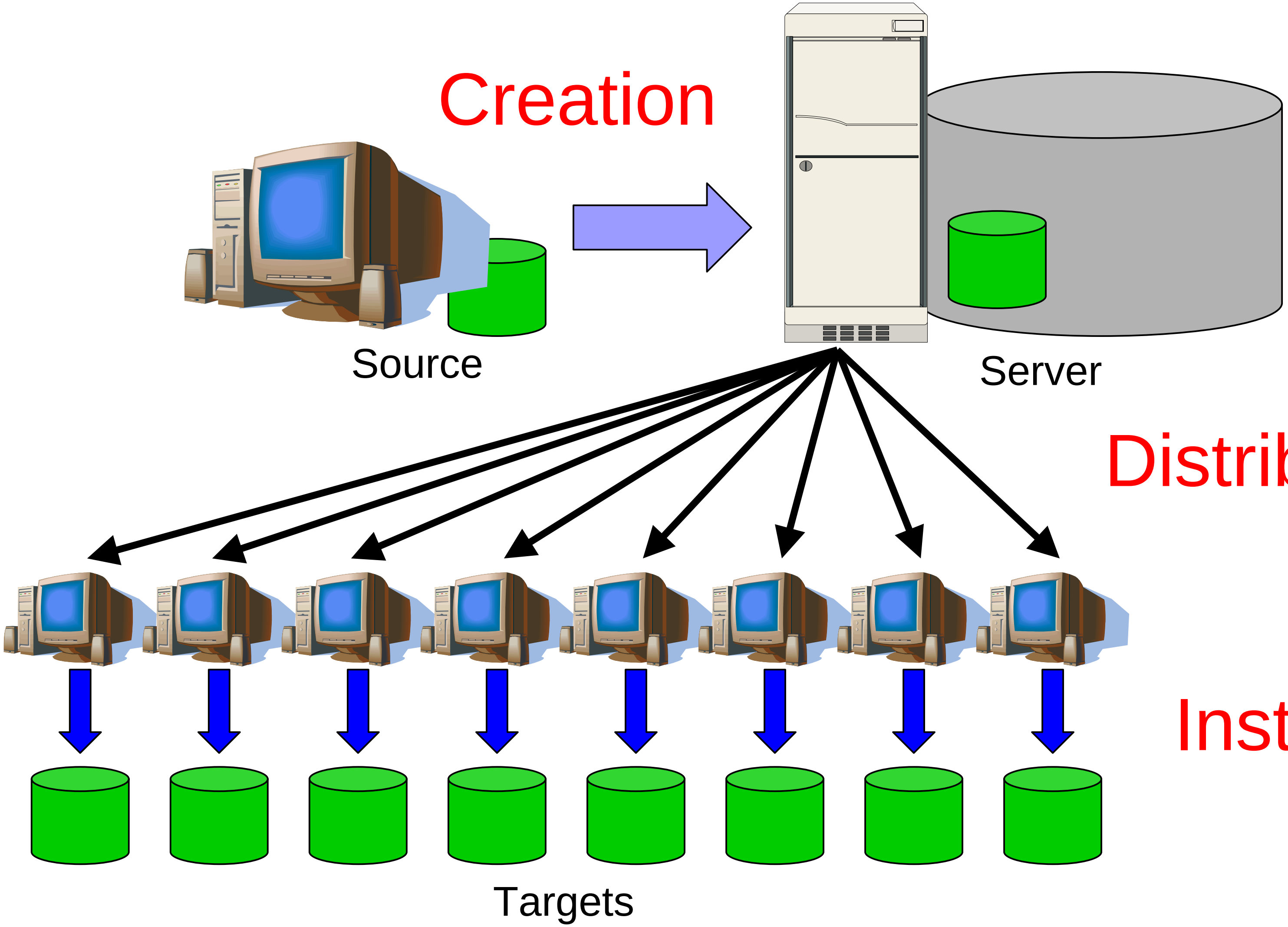
The Emulab Environment

- Network testbed for emulation
  - Cluster of 168 PCs 100Mbps Ethernet LAN
- Users have full root access to nodes
- Configuration stored in a central database
  - Fast reloading encourages aggressive experiments
  - Swapping to free idle resources
- Custom disk images
- Frisbee in use 18 months, loaded > 60,000 disks

Disk Imaging Unique Features

- General and Versatile
  - Does not require knowledge of filesystem
  - Can replace one filesystem type with another
- Robust
  - Old disk contents irrelevant
- Fast

Disk Imaging Tasks



Creation

Source

Server

Distribution

Installation

Targets

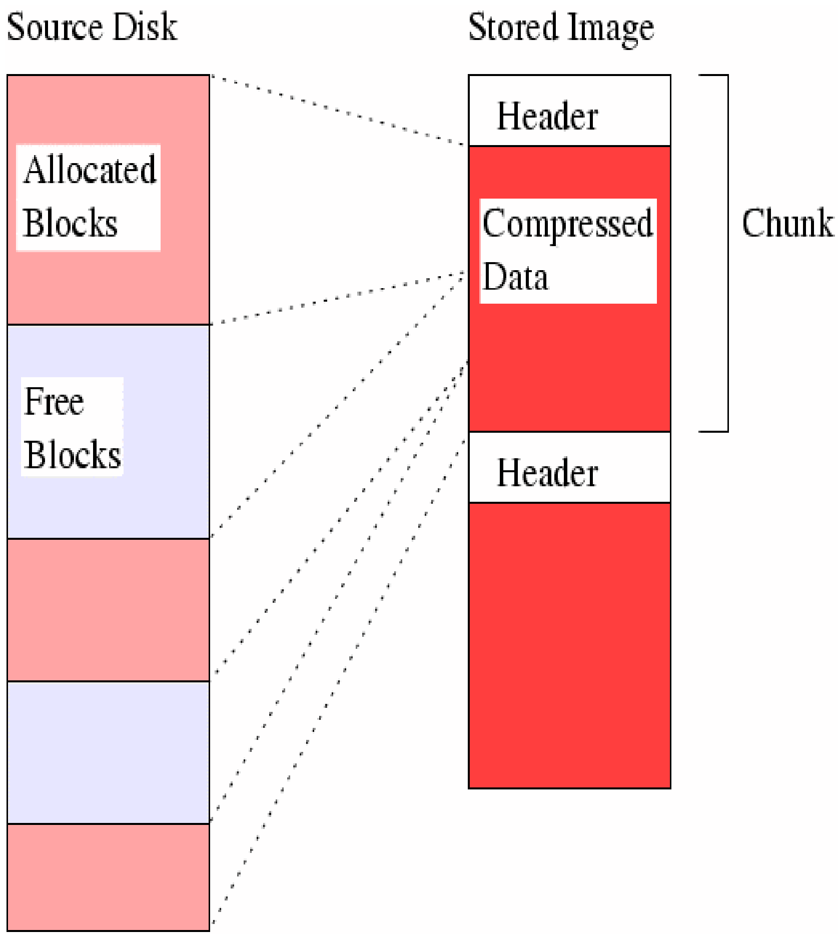
Key Design Aspects

- Domain-specific data compression
- Two-level data segmentation
- LAN-optimized custom multicast protocol
- High levels of concurrency in the client

Image Creation

- Segments images into self-describing “chunks”
- Compresses with zlib
- Can create “raw” images with opaque contents
- Optimizes some common filesystems
  - ext2, FFS, NTFS
  - Skips free blocks

Image Layout



- Chunk logically divided into 1024 blocks
- Medium-sized chunks good for
  - Fast I/O
  - Compression
  - Pipelining
- Small blocks good for
  - Retransmits

Image Distribution Environment

- LAN environment
  - Low latency, high bandwidth
  - IP multicast
  - Low packet loss
- Dedicated clients
  - Consuming all bandwidth and CPU OK

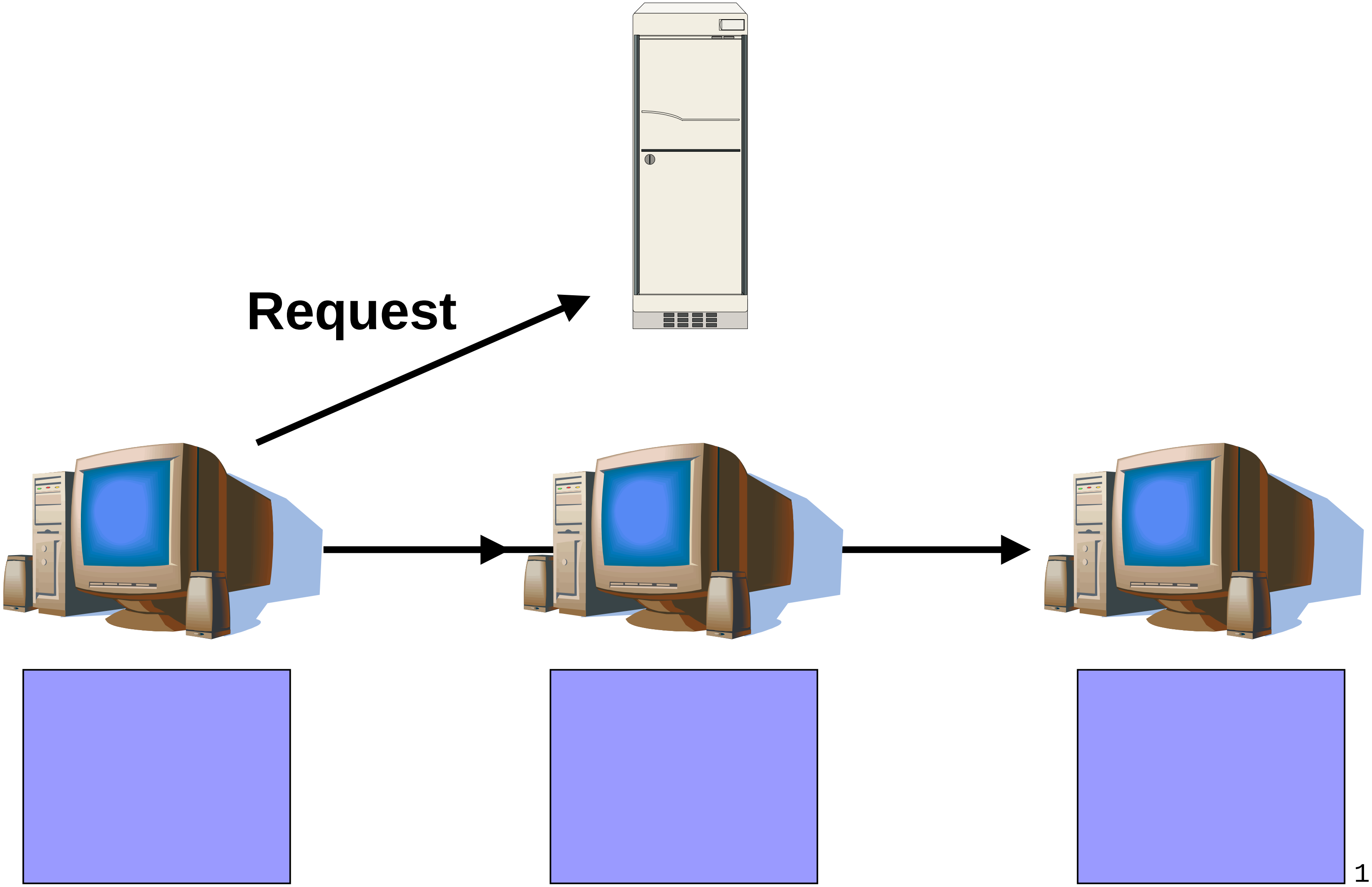
Custom Multicast Protocol

- Receiver-driven
  - Server is stateless
  - Server consumes no bandwidth when idle
- Reliable, unordered delivery
- “Application-level framing”
- Requests block ranges within 1MB chunk

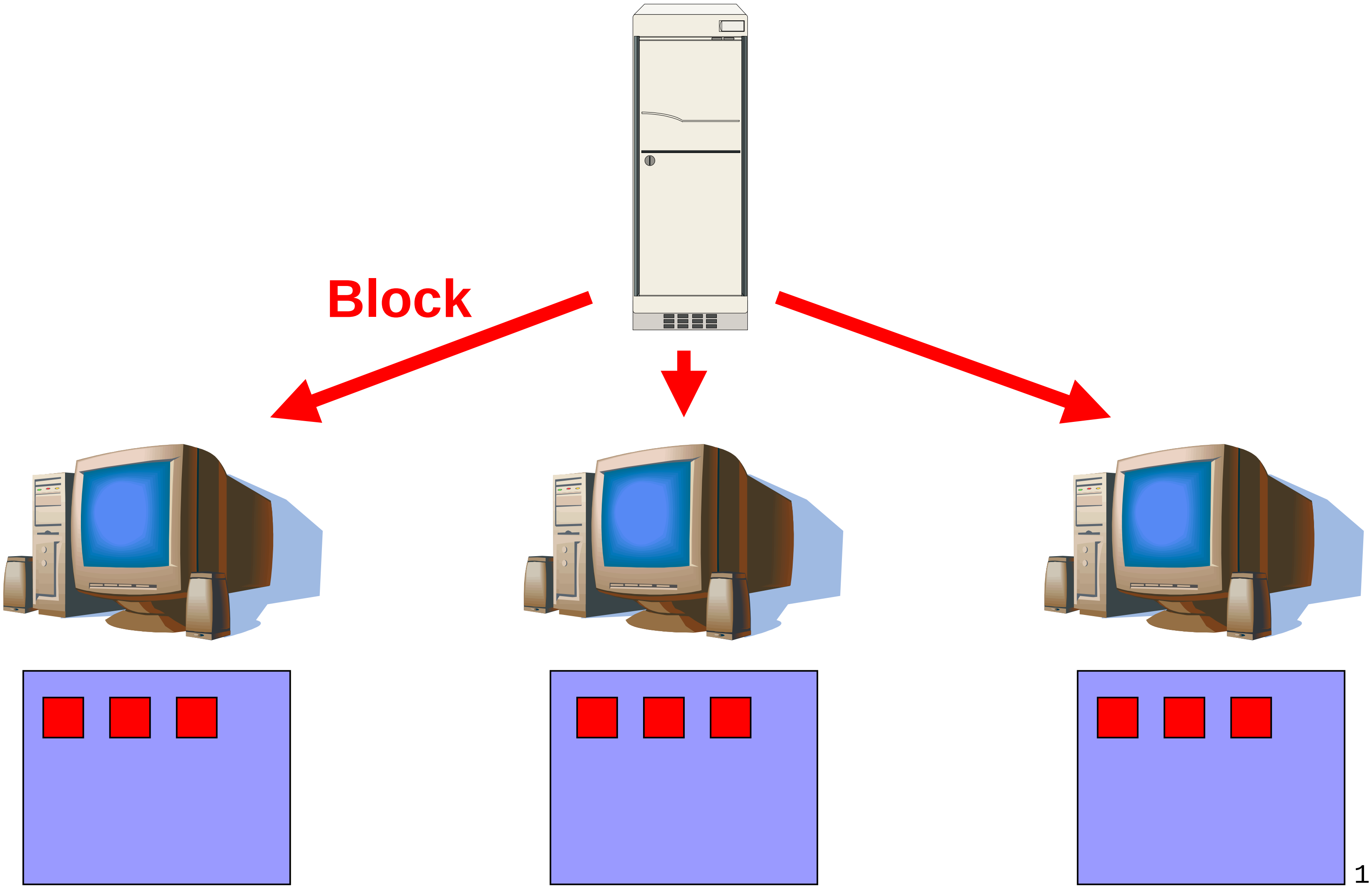
Client Operation

- Joins multicast channel
  - One per image
- Asks server for image size
- Starts requesting blocks
  - Requests are multicast
- Client start not synchronized

Client Requests



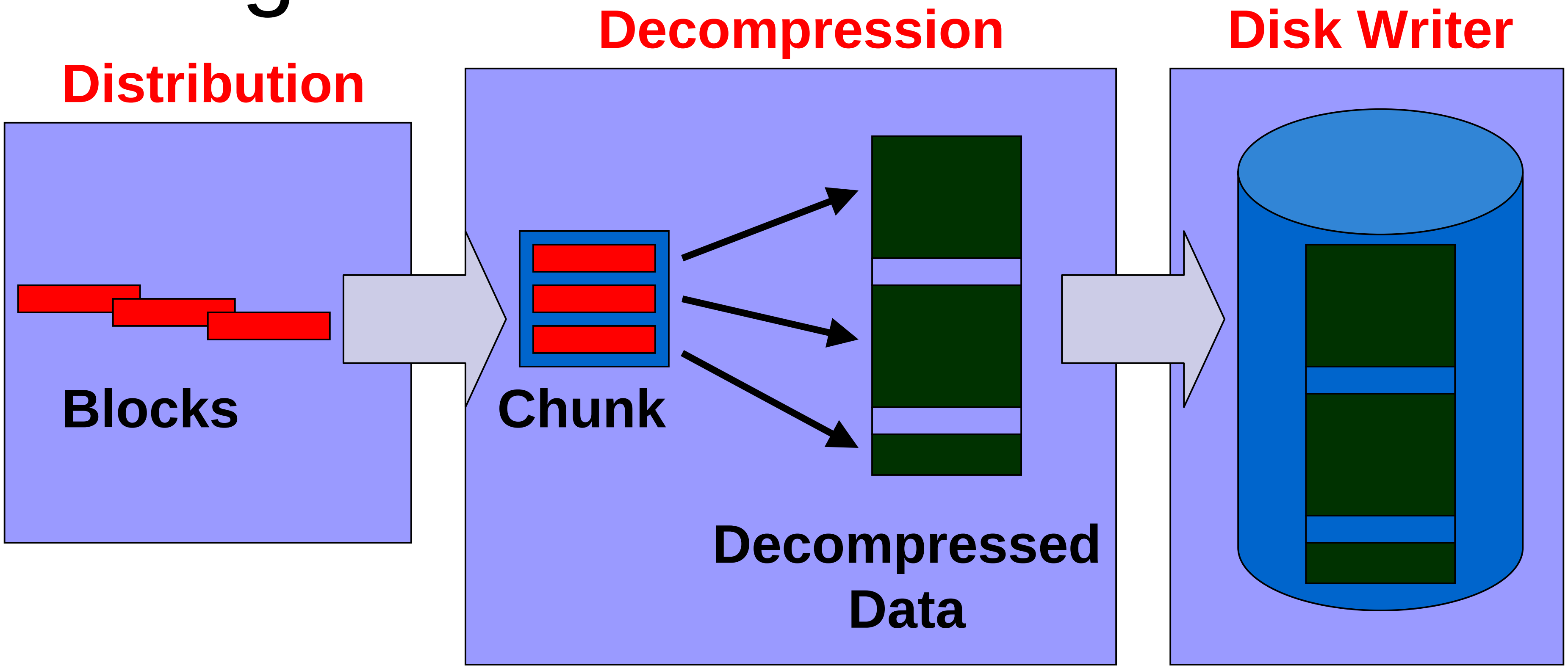
Client Requests



Tuning is Crucial

- Client side
  - Timeouts
  - Read-ahead amount
- Server side
  - Burst size
  - Inter-burst gap

Image Installation



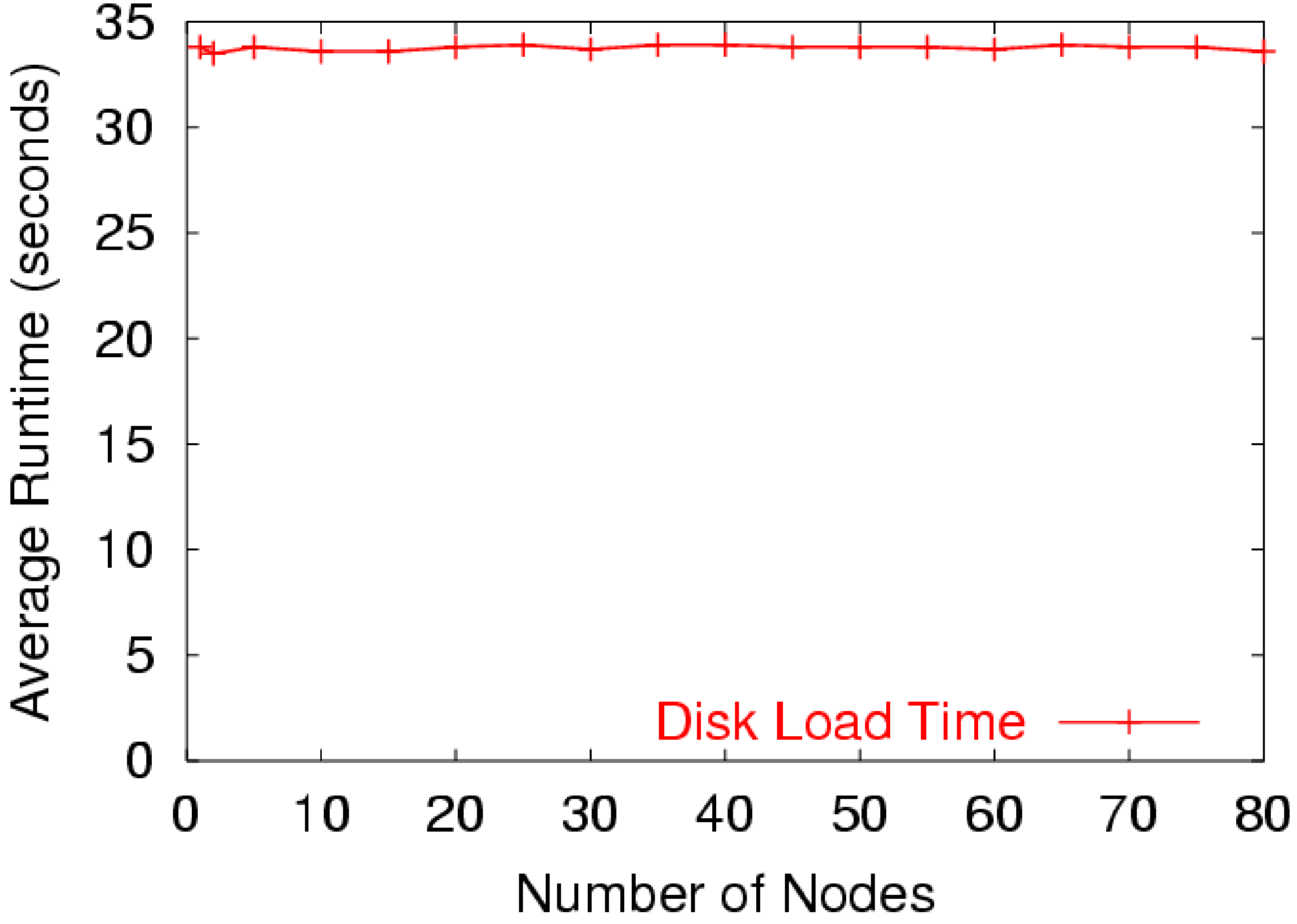
- Pipelined with distribution
  - Can install chunks in any order
  - Segmented data makes this possible
- Three threads for overlapping tasks
- Disk write speed the bottleneck
- Can skip or zero free blocks

Evaluation

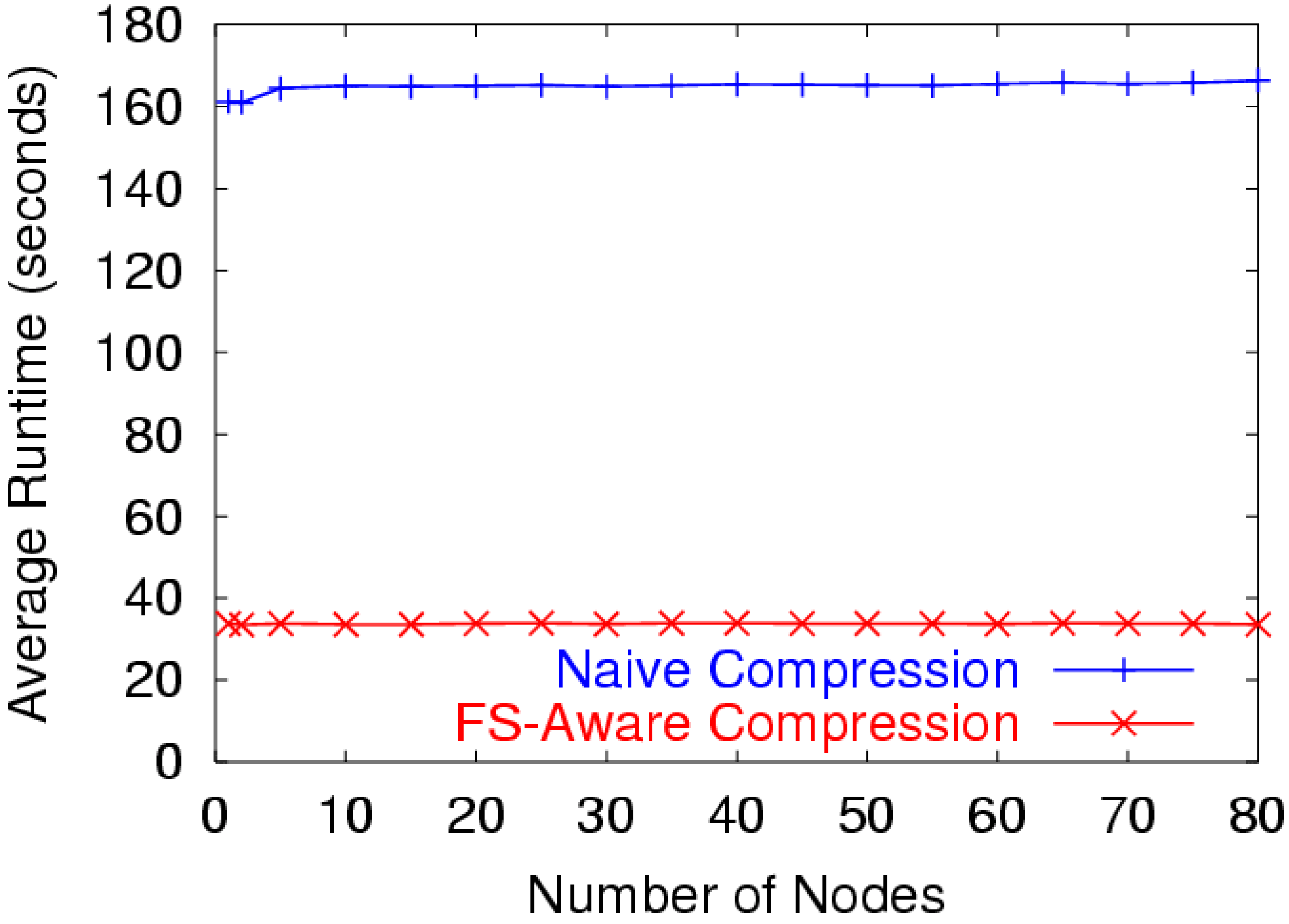
Performance

- Disk image
  - FreeBSD installation used on Emulab
  - 3 GB filesystem, 642 MB of data
  - 80% free space
  - Compressed image size is 180 MB
- Client PCs
  - 850 MHz CPU, 100 MHz memory bus
  - UDMA 33 IDE disks, 21.4 MB/sec write speed
  - 100 Mbps Ethernet, server has Gigabit

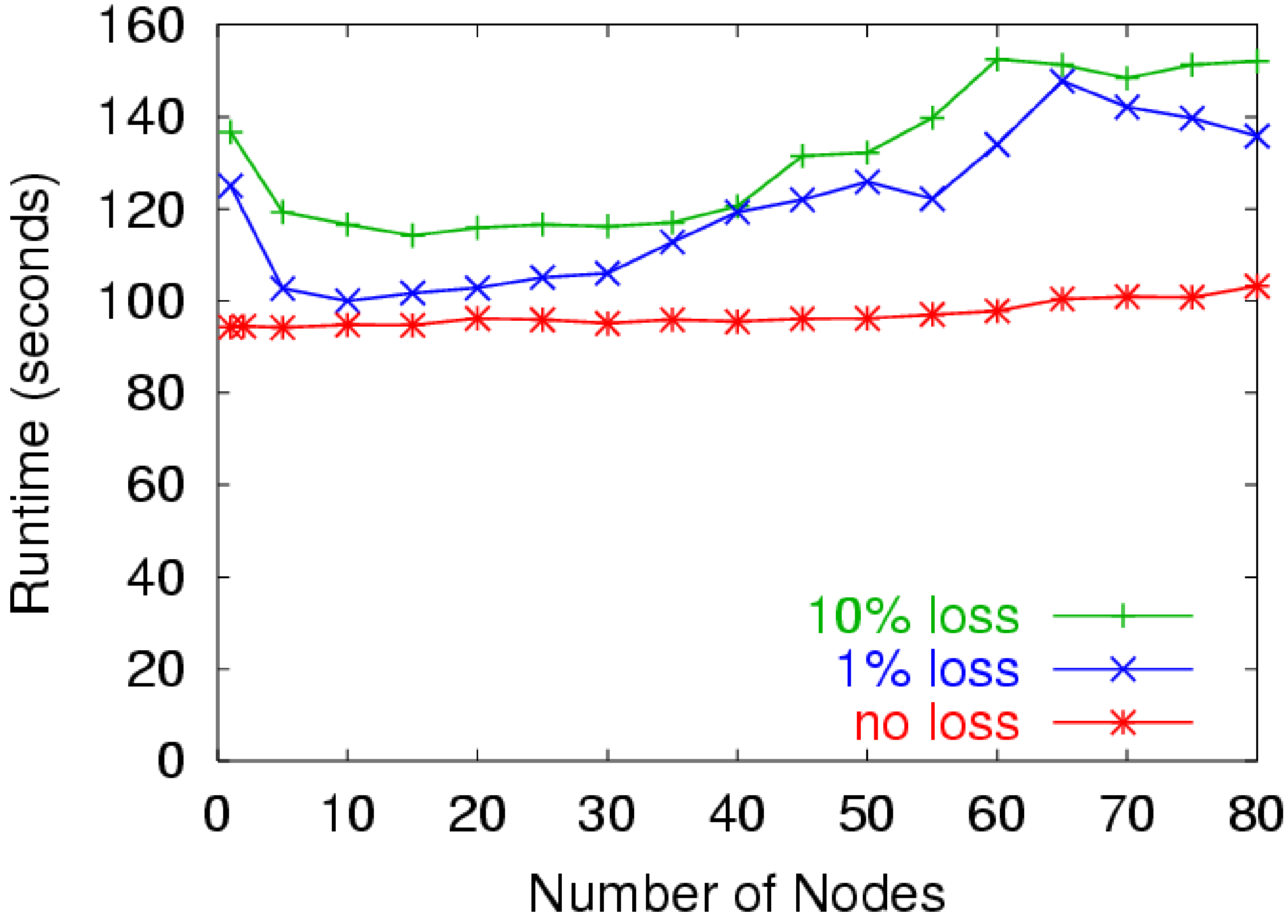
Speed and Scaling



FS-Aware Compression



Packet Loss

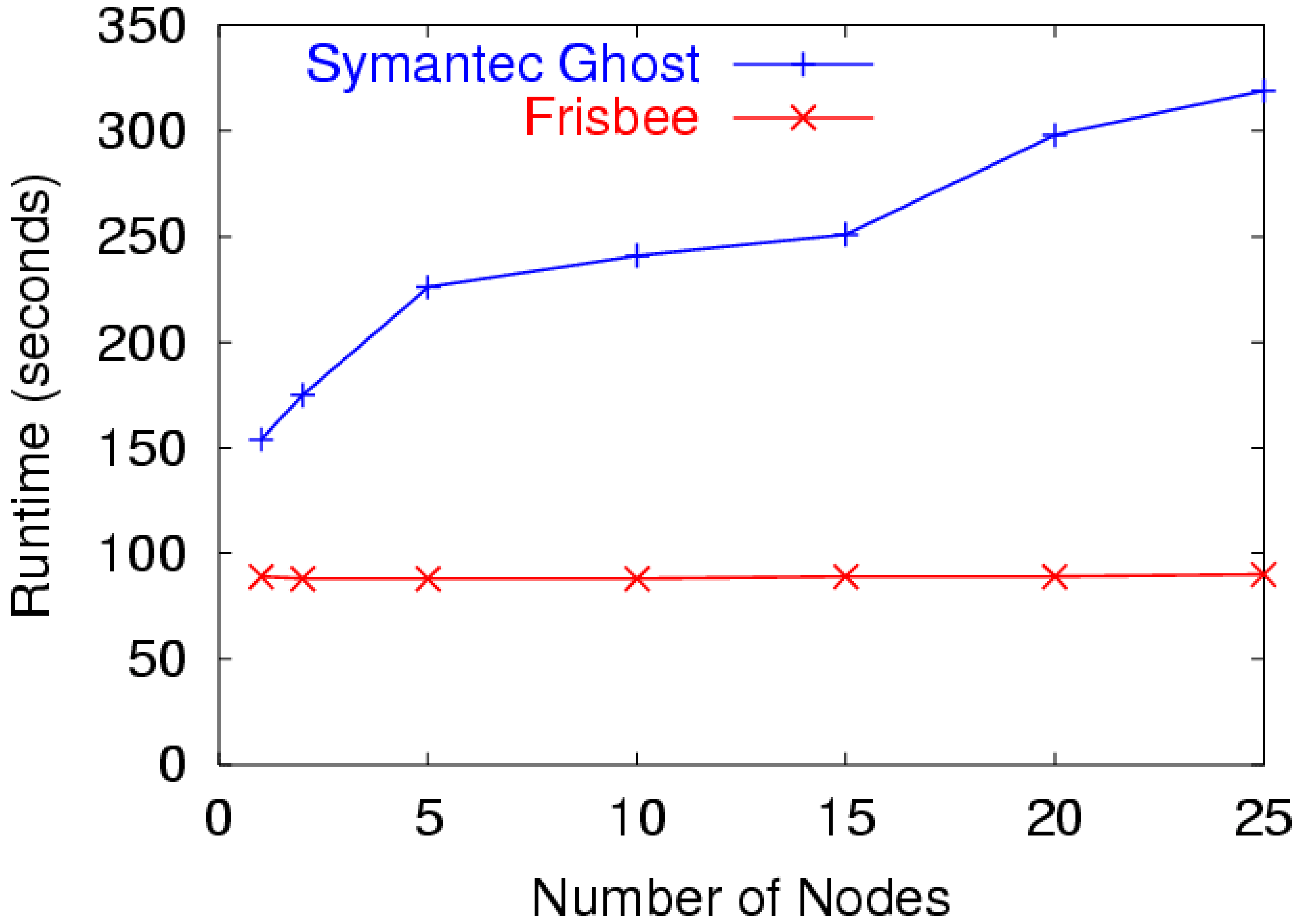


Runtime (seconds)
160
140
120
100
80
60
40
20
0
0
10
20
30
40
50
60
70
80
Number of Nodes
10% loss
1% loss
no loss

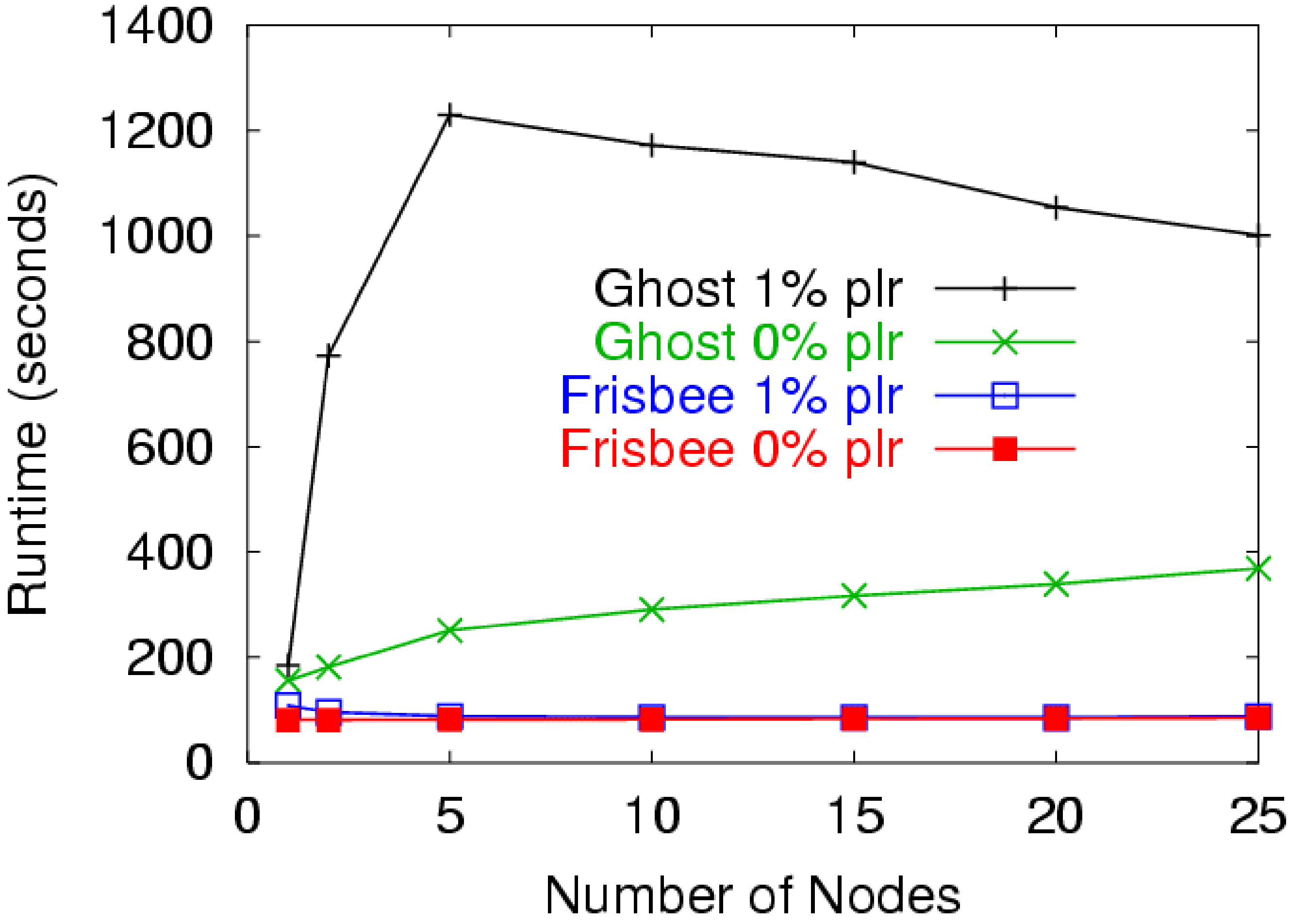
Related Work

- Disk imagers without multicast
  - Partition Image [www.partimage.org]
- Disk imagers with multicast
  - PowerQuest Drive Image Pro
  - Symantec Ghost
- Differential Update
  - rsync 5x slower with secure checksums
- Reliable multicast
  - SRM [Floyd ’97]
  - RMTP [Lin ’96]

Comparison to Symantec Ghost



Ghost with Packet Loss



How Frisbee Changed our Lives (on Emulab, at least)

- Made disk loading between experiments practical
- Made large experiments possible
  - Unicast loader maxed out at 12
- Made swapping possible
  - Much more efficient resource usage

The Real Bottom Line

“I used to be able to go to lunch while I loaded a disk, now I can’t even go to the bathroom!”

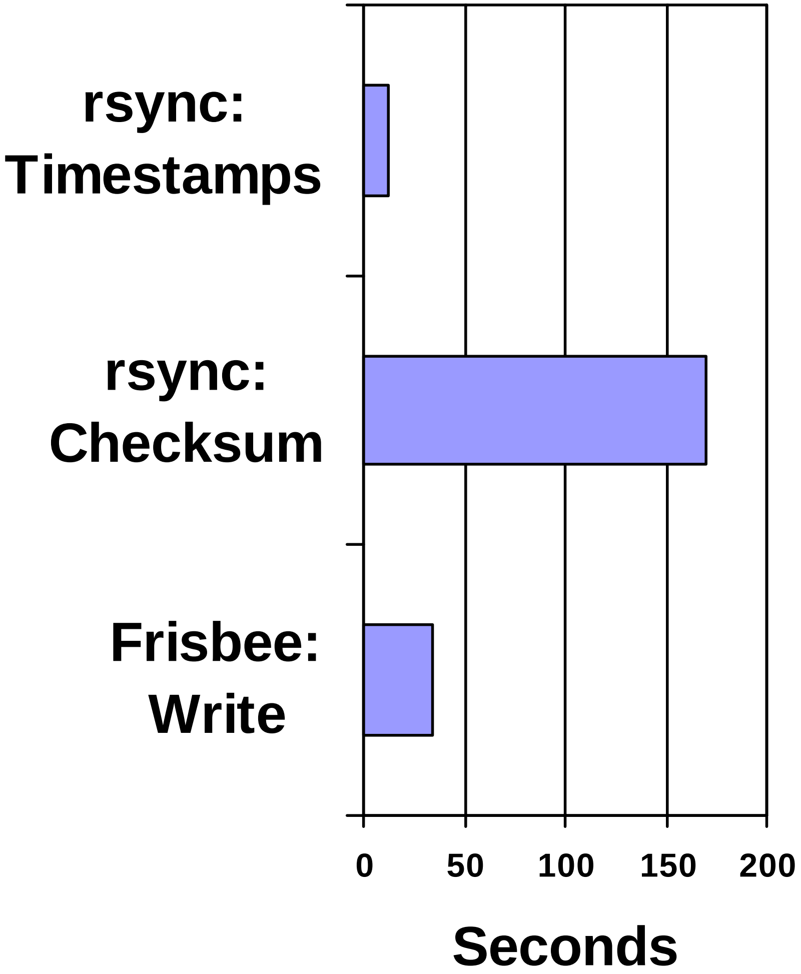
- Mike Hibler (first author)

Conclusion

- Frisbee is
  - Fast
  - Scalable
  - Proven
- Careful domain-specific design from top to bottom is key

Source available at www.emulab.net

Comparison to rsync

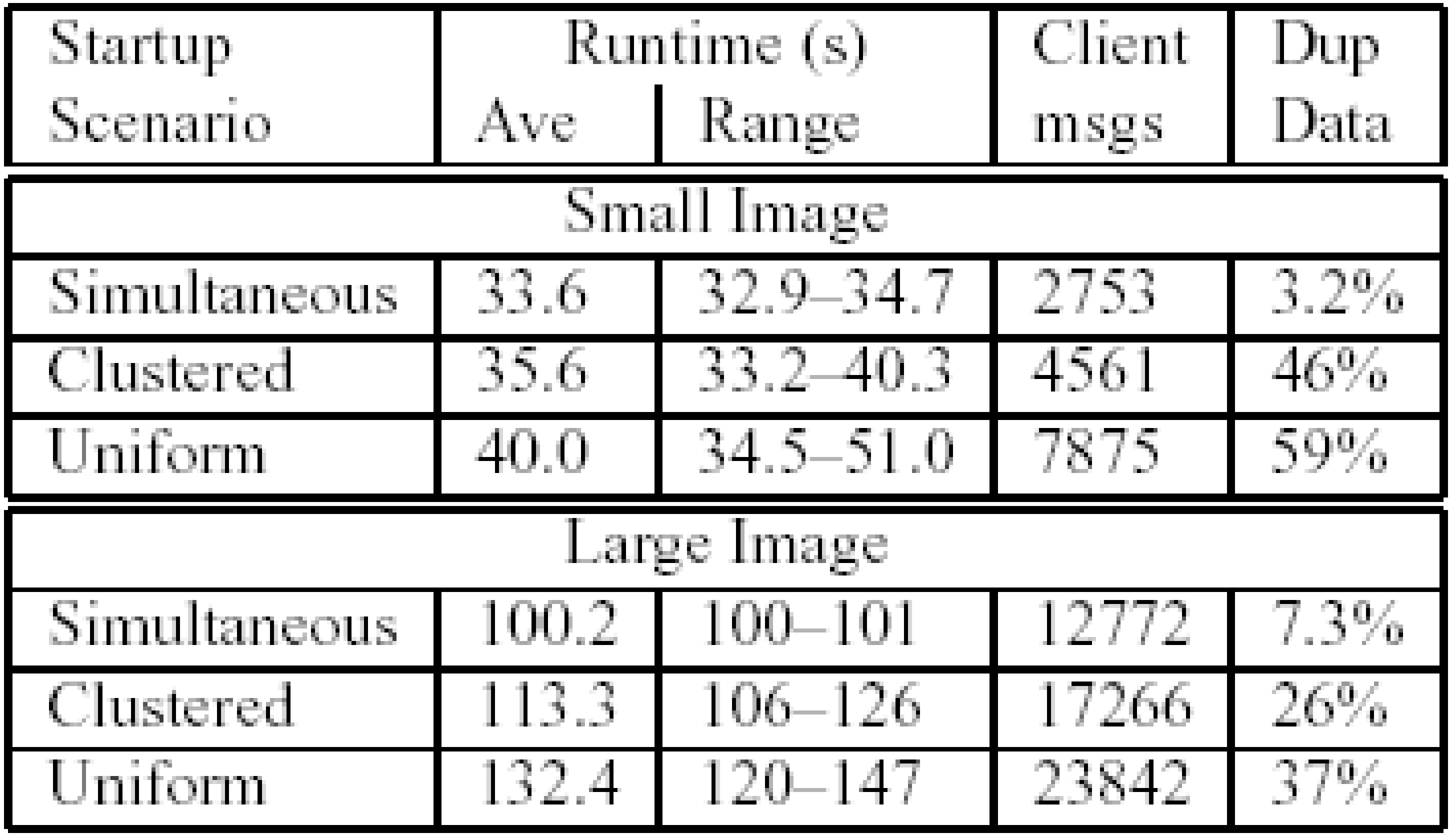


- Timestamps not robust
- Checksums slow
- Conclusion: Bulk writes beat data comparison

How to Synchronize Disks

- Differential update - rsync
  - Operates through filesystem
  - + Only transfers/writes changes
  - + Saves bandwidth
- Whole-disk imaging
  - Operates below filesystem
  - + General
  - + Robust
  - + Versatile
- Whole-disk imaging essential for our task

Image Distribution Performance: Skewed Starts

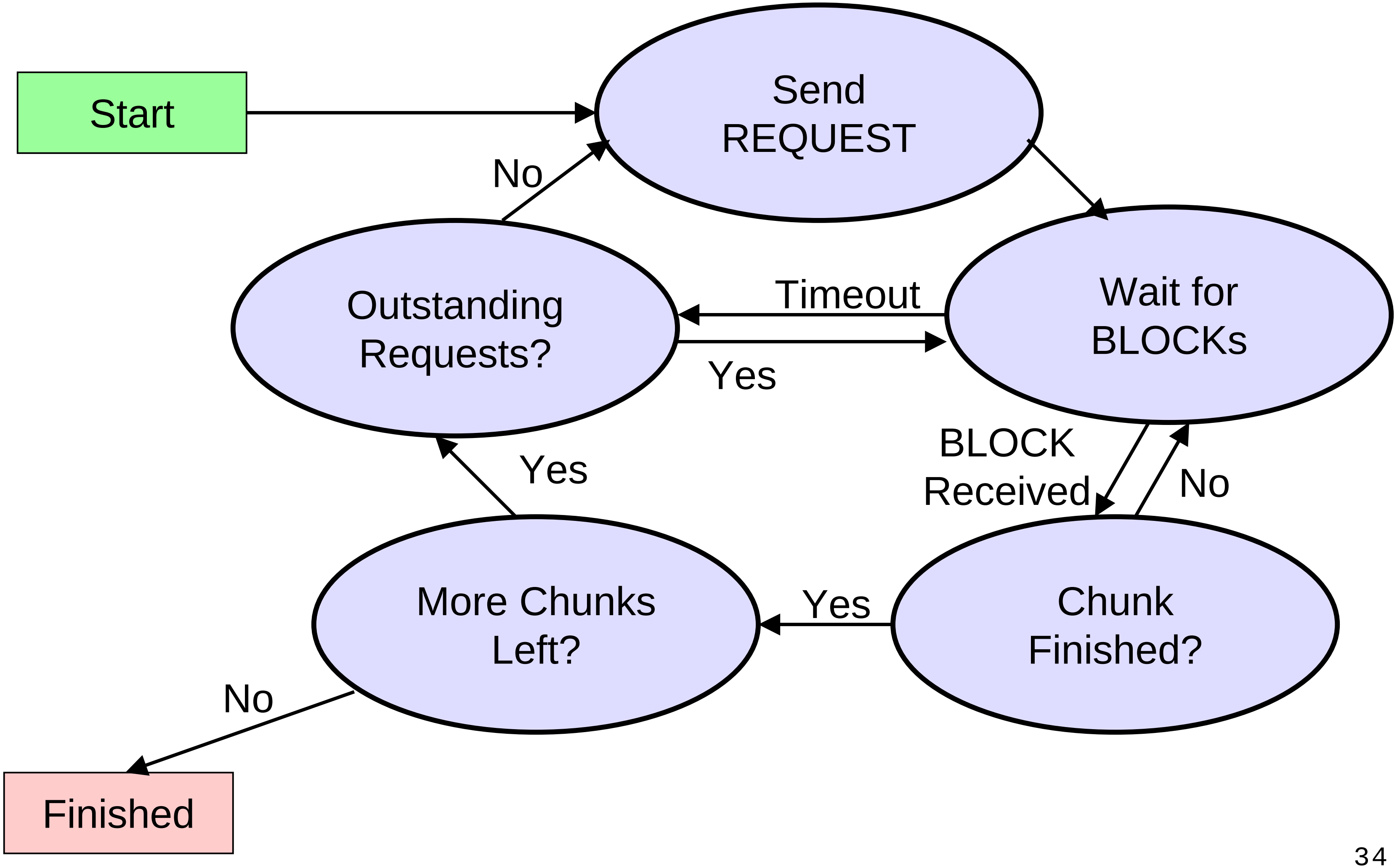


Startup	Runtime (s)		Client	Dup
Scenario	Ave	Range	msgs	Data
Small Image				
Simultaneous	33.6	32.9–34.7	2753	3.2%
Clustered	35.6	33.2–40.3	4561	46%
Uniform	40.0	34.5–51.0	7875	59%
Large Image				
Simultaneous	100.2	100–101	12772	7.3%
Clustered	113.3	106–126	17266	26%
Uniform	132.4	120–147	23842	37%

Future

- Server pacing
- Self tuning

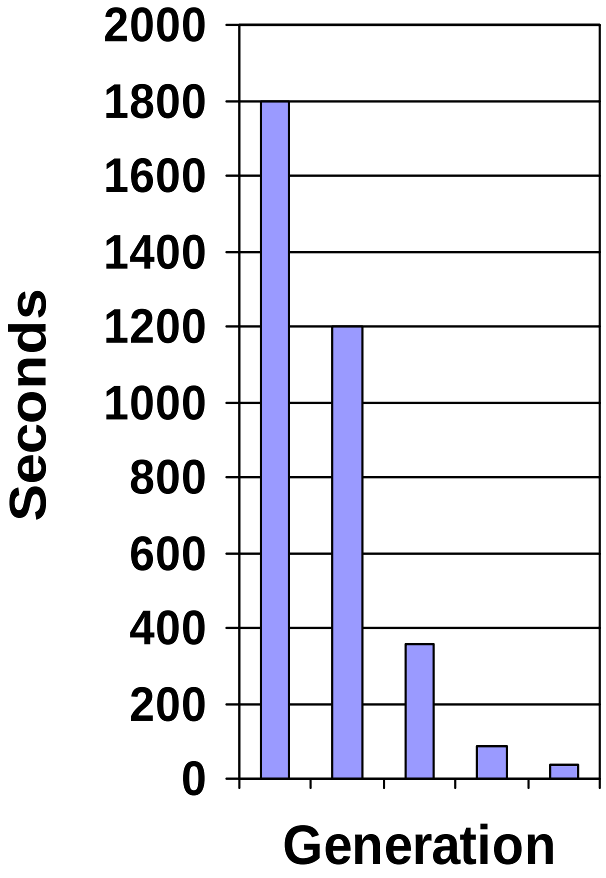
The Frisbee Protocol



The Evolution of Frisbee

- First disk imager: Feb, 1999
- Started with NFS distribution
- Added compression
  - Naive
  - FS-aware
- Overlapping I/O
- Multicast

30 minutes down to 34 seconds!


Seconds
Generation